# Specifications of DVD Content Scramble System

# for

# Authenticator on Decoder Card

Version 0.9

October, 1996

Matsushita Electric Industrial Co., Ltd.

Table1. General Specifications

| Content Scramble | |
|---|---|
| Object of scramble | Packed data (as defined in ISO/IEC 13818-1) |
| Scramble/ Non-scramble | selectable Video Title Set (VTS) by VTS (as defined in "DVD Specifications for Read-Only disc ; Part3 Video Specifications) |
| Scramble/Encryption process | AV Data is scrambled using Title Key.<br>Title Key is encrypted using Disc Key.<br>Disc Key is encrypted using Master Key. |
| Descramble/Decryption process | Disc Key is reproduced using Master Key.<br>Title Key is reproduced using Disc Key.<br>AV Data is reproduced using Title Key. |
| Key Selection | Master Key is predetermined for DVD Content Scramble System.<br>Disc Key is chosen for each Side of Disc.<br>Title Key is chosen for each VTS. |
| Algorithms | Secret Disc Key encryption/decryption algorithm<br>Secret Title Key encryption/decryption algorithm<br>Secret AV Data scramble/descramble algorithm |
| Security Management | Disc Key and Title Key encryption is conducted in Key Protection Module.<br>AV Data scrambling is conducted by a black box.<br>Disc Key and Title Key decryption and AV Data descrambling is conducted in a protected module in DVD Video Player or Decoder card. |
| Bus-Authentication and Bus-Encryption | |
| Authentication | Challenge-response mutual authentication |
| Key sharing | As the result of authentication, DVD-ROM Drive and Decoder Card have the same time variable Bus Key. |
| Bus-Encryption/Decryption | Secured Disc Key data and Encrypted Title Key are encrypted/decrypted using the time variable Bus Key. |
| Algorithms | - Secret oneway functions for mutual authentication<br>- Secret oneway function for generating time variable Bus Key<br>- Authentication Control Code, which is chosen for each side of disc, selects these oneway functions. |

1.4 Terminologies (T.B.D)

Scramble : the process of disguising a AV Data as to hide its substance

Descramble : inverse of Scramble

Encryption : the process of disguising a special data (for example a key) as to hide its substance

Decryption : inverse of Encryption

Key : the data for which used in Scramble/Descramble or Encryption/Decryption

## 2 System Architecture

2.1 Framework of "DVD Content Scramble System"

*Encryption Keys*

- Three keys are used for each prerecorded work: a Master Key, a Disc Key, and a Title Key. An Authentication Control Code is also used for each prerecorded work. The Authentication Control Code is utilized only in the computer environment, as a part of the internal authentication process during transmission of the other keys to the descrambler and associated decoder.

*Encryption /Decryption process*

- For each work that a copyright owner wishes to subject to this system, encryption would proceed as follows:

(1) the content would be scrambled using Title Key and the algorithms and related system technology. The content owner would determine Title Key on its own. The scrambling would be done at disc replication facilities.

(2) Title Key would be encrypted using Disc Key. Encrypted Title Key would be recorded in the Sector Header area, which is a "hidden" area. ("Hidden" here and elsewhere in this part means that the area is not readily accessed by the user.) This step is done by using Title Key Encryption Module.

(3) Disc Key would be converted to "Secured Disc Key data" - effectively encrypting this key - using Disc Key Protection Module that has Master Keys. The Secured Disc Key data would be written in the hidden area of the disc called the "Lead-in Area".

(4) Authentication Control Code is provided on the disc for use in the authentication process applicable to DVD-ROM devices connected in the computer environment. This Code is used to transmit the other keys to the descrambler and associated decoder and is independent of the content scrambling/encryption process itself.

NOTE: The security of the scrambling/encryption process used by studios and disc replication facilities is maintained through both the separate application of keys not necessarily held by the same party and the sealing of the Data Scramble, Title Key

Encryption, and Disc Key Protection Logic function in hardware.

- For playback by DVD Video Player (stand-alone) device, descrambling would be accomplished through the following process:
  1) Disc Key Recovery Logic in the descrambler would read Secured Disc Key data from the hidden Lead-in Area, recovering Disc Key;
  2) Descrambler would then read (decrypt) Encrypted Title Key from the hidden Sector Header;
  3) Descrambler would then descramble the audio-video data in real time for playback.

  NOTE: (1) DVD Video Player is expected to have decryption capability; (2) All descrambling and decoding functions can be performed either in hardware or through the use of tamper resistant software.

- Play back though a DVD-ROM drive and DVD Audio/Video Decoder Card would be the same as for stand-alone players except that there is an additional step prior to actual descrambling. Here, DVD-ROM drive and DVD Audio/Video Decoder Card query each other in a bi-directional "dialogue" to verify that both are authorized to send and receive the keys and scrambled data. If the query succeeds and the transfer is authorized, the keys are encrypted and transmitted from the drive to the descramble/decoder.



*Mechanisms*

DVD Content Scramble System is constructed by the following mechanisms :

(a) Encryption of Disc Key

This mechanism is contained in Disc Key Protection Module, and encrypts Disc Key by Master Keys.

(b) Decryption of Disc Key

This mechanism is contained in Descrambler Module at DVD Video Player and Decoder Card, and decrypts Disc Key by Master Key.

(c) Encryption of Title Key

This mechanism is contained in Title Key Encryption Module, and encrypts Title Key by Disc Key.

(d) Decryption of Title Key

This mechanism is contained in Descrambler Module at DVD Video Player and Decoder Card, and decrypts Title Key by Disc Key.

(e) Scrambling of AV Data

This mechanism is contained in Scrambler Module at Disc Formatter, and scrambles AV Data by Title Key.

(f) Descrambling of AV Data

This mechanism is contained in Descrambler at DVD Video Player and Decoder

Card, and descrambles AV Data by Title Key.

(g) Authentication Mechanism

This mechanism is contained in Authenticator Module at DVD-ROM Drive and Decoder Card, and authenticates DVD-ROM Drive and Decoder Card via PC buses in PC system

## 2.2 Architecture of "Content Publishing System"

"Content Publishing System" includes Content Provider, Authoring Studio and Disc Formatter.

"Content Publishing System" generates AV contents and scrambles them if necessary, and publishes them as DVD Disc.

The process of "Content Publishing System" is as follows :

(1) Generation of AV Data

- Content Provider generates AV Data as a Video Object Set (VOBS).
- Copyright Owner (possibly Content Provider) decides freely whether the title of AV Data shall be scrambled or not.

(2) Title Key and Disc Key

- When Copyright Owner prefers to scramble the title, Copyright Owner shall select the "Title Key" on a Video Title Set (VTS) by VTS basis.
- When a DVD disc has one or more scrambled VTS, Copyright Administrator (Movie Company or other entity) selects "Disc Key" on a Side by Side basis.
- Disc Key is set to Disc Key Protection Module. Title Key Set is set to Title Key Encryption Module.

(3) Encryption of Disc Key

- Disc Key Protection Module encrypts Disc Key using predetermined "Master Keys" of all descrambler chip manufacturers and generates "Secured Disc Key data" .
- The encryption algorithm is denoted as EncDK.
- Secured Disc Key data is recorded in Lead-in area of Disc.

(4) Encryption of Title Key

- Title Key Encryption Module encrypts each Title Key of Title Key Set using above Disc Key.
- The encryption algorithm is denoted as EncTK.
- Encrypted Title Key is recorded in the Copyright Management Information field of all sectors of a scrambled file.

(5) Authentication Control Code

- "Authentication Control Code" shall be decided for each side of disk, which can

Bus-Encryption/Decryption by BK is as follows :

(1) On Disc Inserted, DVD-ROM Drive encrypts Secured Disc Key data by using BK and sends them to Decoder Card. Decoder Card descrambles them by using BK, and gets the Secured Disc Key data.

(2) Before a playback of VTS, DVD-ROM Drive encrypts Encrypted Title Key by using BK, and sends them to Decoder Card with Scrambled AV Data. Decoder Card decrypts Bus-Encrypted Encrypted Title Key by using BK, and gets Encrypted Title Key and scrambled AV Data. After that, process continues to step (1) in 2.4

If the mutual authentication process is successful, Scrambled AV Data on DVD-ROM Drive is sent to Decoder Card.

To sum up, Fig.4 shows the Architecture of "DVD Video Playback system on PC".

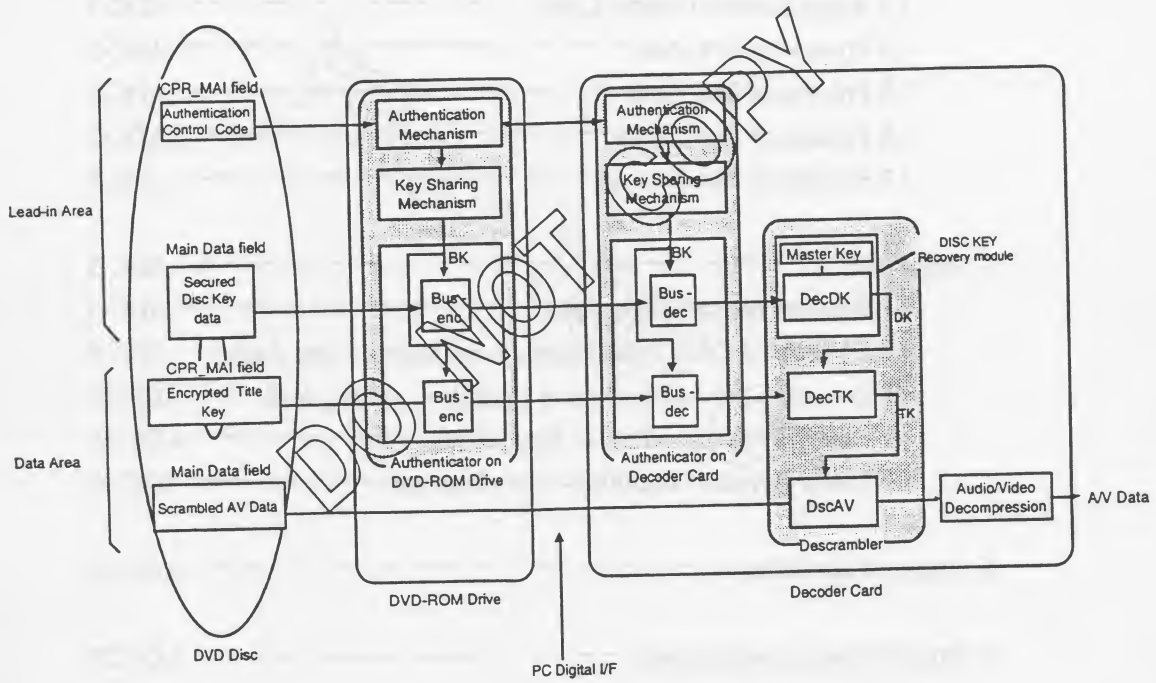


Fig.4 Architecture of "DVD Video Playback system on PC"

## 3. Nominative References

1) DVD Specifications for Read-Only Disc (Version 1.0) : August 1996

2) ISO/IEC 13818 : 1994
Information Technology-generic coding of moving pictures and associated audio.

# 1. General

## 1.1 Scope

This part defines specification of DVD Content Scramble System for Authenticator on Decoder Card.

This part defines functions and algorithms in Authenticator. And these are used for Bus-Authentication and Bus-Decryption mechanism.

- Objectives of Bus-Authentication and Bus-Decryption are as follows :
  - Bus-Authentication
    To prevent digital-to-digital copy on personal computer environment.
  - Bus-Decryption
    To prevent the illegal-replacement after the mutual authentication.

To realize these objectives Decoder Card has the following mechanisms :

- Authentication mechanism : to determine that Decoder Card is 'authentic' and to let Decoder Card determine that DVD-ROM Drive is 'authentic'.
- Key sharing mechanism : to share a time variable key used for Bus-Decryption.
- Bus-Decryption mechanism : to decrypt Bus-Encrypted Secured Disc Key data and Bus-Encrypted Encrypted Title Key using by the shared time variable key.

In addition to these mechanism, Decoder Card has Descrambler mechanism.

Fig.1 shows the architecture of Decoder Card.

may be changed by using Authentication Control Code.

1.2 General Specifications

Table1 shows the general specifications for Authenticator on Decoder Card.

Table1. General Specifications for Authenticator on Decoder Card

| | |
|---|---|
| Random number Generation | [Output] Drv_Chal (80 bits) : Authenticator_on_Decoder's Challenge data for Drive authentication |
| Drv_Auth | [Input] Authentication Control Code (5 bits)<br>[Output] Drv_Ref (40 bits) : Authenticator_on_Decoder's Reference data for Drive authentication |
| Dec_Auth | [Input] Dec_Chal (80 bits) : Authenticator_on_Drive's Challenge data for Decoder authentication<br>[Input] Authentication Control Code (5 bits)<br>[Output] Dec_Res (40 bits) : Authenticator_on_Decoder's Response data for Decoder authentication |
| Key_Share | [Input] Authentication Control Code (5 bits)<br>Calculate a time-variable key (BK) after mutual authentication, and store it without appearing outside |
| Bus-Decrypt | <On Disc Inserted><br>[Input] Bus-Encrypted Secured Disc Key data (2048 bytes)<br>[Output] Connect to Descrambler without appearing outside<br><Before a playback of VTS><br>[Input] Bus-Encrypted Encrypted Title Key (40 bits)<br>[Output] Connect to Descrambler without appearing outside |



1.3 Terminologies and Notation (T.B.D)

- Vector assignment :

For example, the equation

$ia(39)...ia(0) = ib(39)...ib(0) \oplus ic(39)...ic(0)$

means the following equations are satisfied :

$ia(39) = ib(39) \oplus ic(39)$

$ia(38) = ib(38) \oplus ic(38)$

....

$ia(0) = ib(0) \oplus ic(0)$.

Note that each ia(i), ib(i) and ic(i) for i=39-0 means a bit.

3.2 Authentication Key

Every Authenticator has a universal (common) 40 bits Authentication Key.

All oneway functions (Drv_Auth, Dec_Auth and Key_Share) include the Authentication Key, and we omit to write it for each function.

3.3 Authentication Control Code

5 bits Authentication Control Code denoted as "ACC" selects which oneway functions of 32 functions for each Drv_Auth(ACC, j)/Dec_Auth(ACC, j)/Key_Share (ACC, j) (ACC=0-31, j is 80 bits data) is used.

3.4 Process of Drv_Auth

Drv_Auth calculates the 40 bits reference data "Drv_Ref" using the 80 bits random number stored internally "Drv_Chal".

3.5 Process of Dec_Auth

Dec_Auth calculates the 40 bits response data "Dec_Res" using a 80 bits challenge data from Decoder Card denoted as "Dec_Chal".

3.6 Process of Key_Share

Key_Share calculates a time variable key BK using the data stored internally after mutual authentication using by Drv_Auth and Dec_Auth.

3.7 Process of Bus-Decrypt

On Disc inserted, Authenticator in Decoder Card decrypts Bus-Encrypted Secured Disc Key data (2048 bytes) from DVD-ROM Drive, using shared BK.

Before a playback of VTS, Authenticator in Decoder Card decrypts Bus-Encrypted Encrypted Title Key data (40 bits) from DVD-ROM Drive, using shared BK.



## 4. Algorithm

4.1 Random number Generation

We do not define the structure of Random number Generation.

We only define the conditions of Random number Generation as follows :

- It may be generated by a real random number generator.
- If it is implemented by pseudo-random number generator, its linear complexity shall be over 41.

4.2 DVD-ROM Drive Authentication algorithm--- Drv_Auth

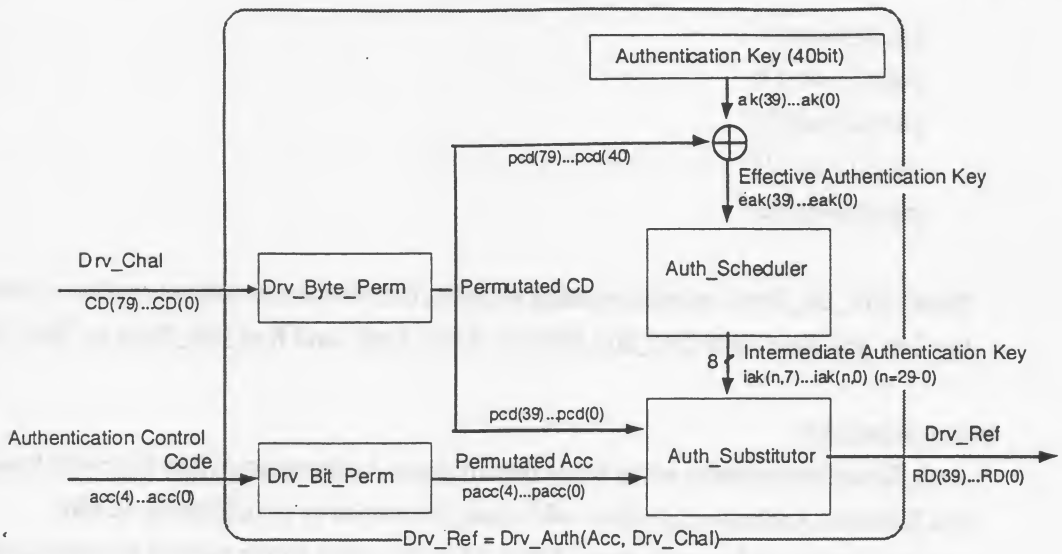


Fig.2 The structure of Drv_Auth

Next we define Drv_Byte_Perm, Drv_Bit_Perm, Auth_Scheduler and Auth_Substitutor in detail.

[Drv_Byte_Perm]

Drv_Byte_Perm permutates 80 bits Drv_Chal (of which each bit is represented as CD(79)...CD(0)) to 80 bits Permutated CD (of which each bit is represented as pcd(79)...pcd(0)). The relation between input and output is as follows :

pcd(79)...pcd(72)=CD(71)...CD(64)
pcd(71)...pcd(64)=CD(39)...CD(32)
pcd(63)...pcd(56)=CD(55)...CD(48)
pcd(55)...pcd(48)=CD(79)...CD(72)
pcd(47)...pcd(40)=CD(23)...CD(16)
pcd(39)...pcd(32)=CD(47)...CD(40)
pcd(31)...pcd(24)=CD(63)...CD(56)
pcd(23)...pcd(16)=CD(7)...CD(0)
pcd(15)...pcd(8)=CD(31)...CD(24)
pcd(7)...pcd(0)=CD(15)...CD(8)

[Drv_Bit_Perm]

Drv_Bit_Perm permutates 5 bits input of Authentication Control Code (of which each bit is represented as acc(4)...acc(0)) to 5 bits Permutated ACC (of which each bit is represented as pacc(4)...pacc(0)). The relation between input bit and output bit is as follows :

- Initialization of carry

  Carry is initialized to "0".

The output bit sequence of Auth_Scheduler is iak(n,7)...iak(n,0) for n=29-0.
Note that iak(29,7) is generated first, iak(29,6)...iak(29,0) is next, and so on. And the last one is iak(0,0).

Fig.3 shows the structure of Auth_Scheduler.

[Auth_Substitutor]

Auth_Substitutor has Permutated CD (that is denoted as pcd(39)...pcd(0)) and Permutated ACC (that is denoted as pacc(4)...pacc(0)) and Intermediate Authentication Key (that is denoted as iak(n,7)...iak(n,0) for n=29-0) as input, and outputs Response Data (that is denoted as RD(39)...RD(0) ).

The Auth_Substitutor algorithm is as follows using by Auth_Sbox explained later :

Note that pacc(4)...pacc(0) enters into each Auth_Sbox, but it is omitted in below not to be complicated.

<1st round>

temp1(39)...temp1(32)=Auth_Sbox(iak(29,7)...iak(29,0), pcd(39)...pcd(32))

temp1(31)...temp1(24)=Auth_Sbox(iak(28,7)...iak(28,0), pcd(31)...pcd(24))
⊕ pcd(39)...pcd(32)

temp1(23)...temp1(16)=Auth_Sbox(iak(27,7)...iak(27,0), pcd(23)...pcd(16))
⊕ pcd(31)...pcd(24)

temp1(15)...temp1(8)=Auth_Sbox(iak(26,7)...iak(26,0), pcd(15)...pcd(8))
⊕ pcd(23)...pcd(16)

temp1(7)...temp1(0)=Auth_Sbox(iak(25,7)...iak(25,0),pcd(7)...pcd(0))
⊕ pcd(15)...pcd(8)

<2nd round>

temp2(39)...temp2(32)=Auth_Sbox(iak(24,7)...iak(24,0),
temp1(39)...temp1(32) ⊕ temp1(7)...temp1(0))

temp2(31)...temp2(24)=Auth_Sbox(iak(23,7)...iak(23,0),
temp1(31)...temp1(24))
⊕ temp1(39)...temp1(32) ⊕ temp1(7)...temp1(0)

temp2(23)...temp2(16)=Auth_Sbox(iak(22,7),iak(22,0), temp1(23)...temp1(16))
⊕ temp1(31)...temp1(24)

temp2(15)...temp2(8)=Auth_Sbox(iak(21,7)...iak(21,0), temp1(15)...temp1(8))
⊕ temp1(23)...temp1(16)

temp2(7)...temp2(0)=Auth_Sbox(iak(20,7)...iak(20,0), temp1(7)...temp1(0))
⊕ temp1(15)...temp1(8)

<3rd round>

temp3(39)...temp3(32)=T(Auth_Sbox(iak(19,7)...iak(19,0),
temp2(39)...temp2(32)⊕temp2(7)...temp2(0)) )
where 'T' box will be explained later.

temp3(31)...temp3(24)=T(Auth_Sbox(iak(18,7)...iak(18,0),
temp2(31)...temp2(24))

⊕ temp5(31)...temp5(24)

RD(15)...RD(8)=Auth_Sbox(iak(1,7)...iak(1,0), temp5(15)...temp5(8))

⊕ temp5(23)...temp5(16)

RD(7)...RD(0)=Auth_Sbox(iak(0,7)...iak(0,0), temp5(7)...temp5(0))

⊕ temp5(15)...temp5(8)

Fig.4 shows the structure of Auth_Substitutor.

[Auth_Sbox]

Auth_Sbox is used 5 times for each round in Auth_Substitutor.

We explain Auth_Sbox algorithm in case that input(j×8+7)...input(j×8) , iak(n,0)...iak(n,7) and pacc(4)...pacc(0) are input and output(j×8+7)...output(j×8) is output for j=4-0.

The Auth_Sbox algorithm is as follows :

(1) The input(i×8+7)...input(j×8) is EORed with iak(n,0)...iak(n,7).

(2) The output of (1) is transformed by Pre-Table. Pre-Table has 8-bits address and 8-bits output, so the memory size of Pre-Table is 256 bytes amount. The content of Pre-Table is shown in Annex-B.

(3) The 7th,5th,4th,2nd and 0th bit of output of (2) are EORed with pacc(4)...pacc(0), respectively.

(4) The output of (3) is transformed by Post-Table. Post-Table has 8-bits address and 8-bits output, so the memory size of Post-Table is 256 bytes amount. The content of Post-Table is shown in Annex-C.

Fig.5 shows the structure of Auth_Sbox.



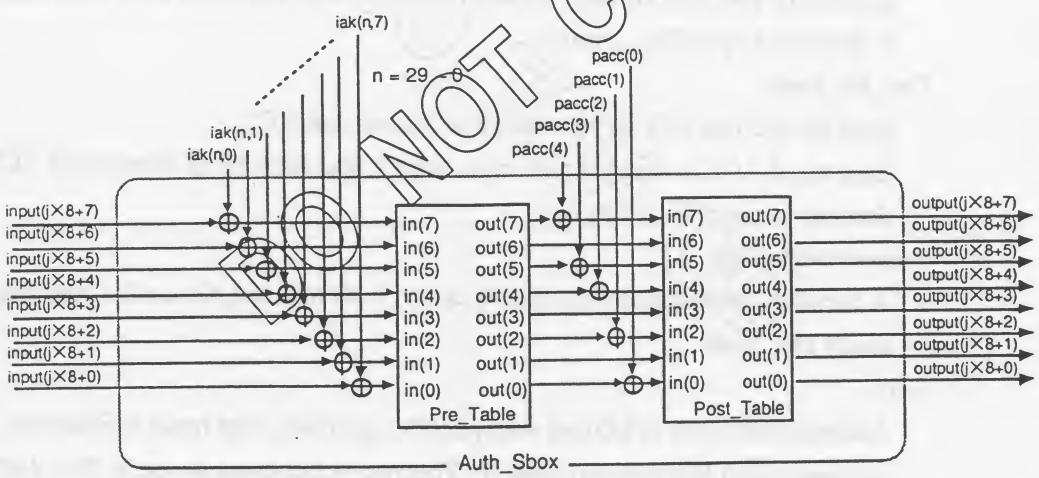


Fig.5 The structure of Auth_Sbox

[T-box]

T-box is a oneway transformation with 8 bits input (IN(0),IN(1),...,IN(7) ) and 8 bits output (OUT(0),OUT(1),..., OUT(7) ).

T-box algorithm is as follows :

OUT(0)=IN(0) ⊕ IN(1) ; "⊕" means EOR

OUT(1)=IN(1) ⊕ IN(2)

OUT(2)=IN(2) ⊕ IN(3)

To sum up, Fig.6 shows the structure of Dec_Auth.

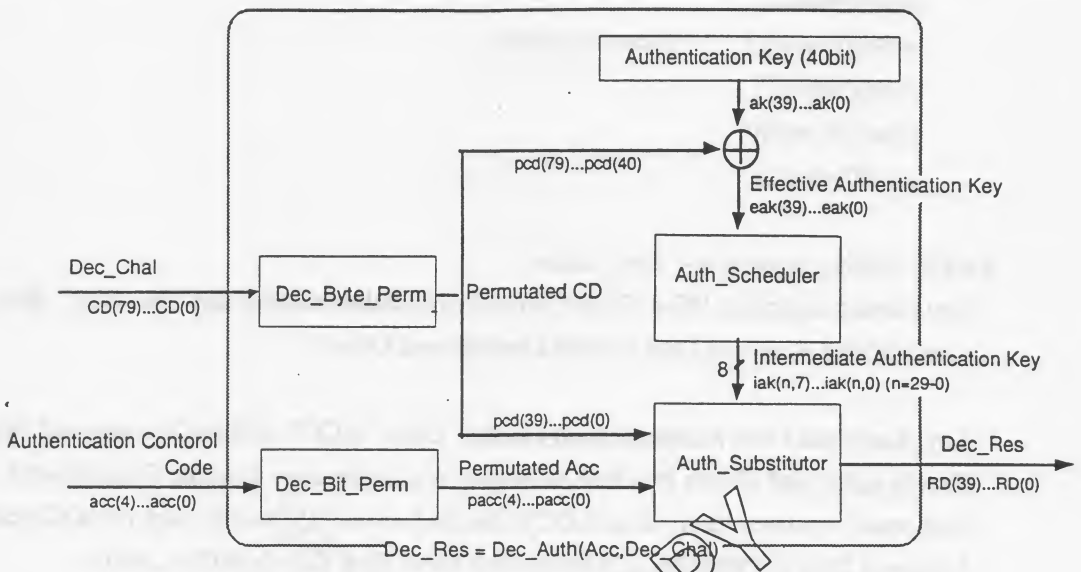


Fig.6 The structure of Dec_Auth

Next we define Dec_Byte_Perm, Dec_Bit_Perm. Note that the other parts are the same as ones in Drv_Auth explained in 4.2.

[Dec_Byte_Perm]

Dec_Byte_Perm permutates 80 bits Drv_Chal ( denoted as CD(79)...CD(0)) to 80 bits Permutated CD (denoted as pcd(79)...pcd(0)). The relation between input and output is as follows :

pcd(79)...pcd(72)=CD(23)...CD(16)
pcd(71)...pcd(64)=CD(7)...CD(0)
pcd(63)...pcd(56)=CD(39)...CD(32)
pcd(55)...pcd(48)=CD(63)...CD(56)
pcd(47)...pcd(40)=CD(47)...CD(40)
pcd(39)...pcd(32)=CD(71)...CD(64)
pcd(31)...pcd(24)=CD(31)...CD(24)
pcd(23)...pcd(16)=CD(79)...CD(72)
pcd(15)...pcd(8)=CD(15)...CD(8)
pcd(7)...pcd(0)=CD(55)...CD(48)

[Dec_Bit_Perm]

Permutated ACC (pacc(4)...pacc(0)) and Intermediate Authentication Key (iak(n,7)...iak(n,0) for n=29-0 ) . This part is the same as one in Drv_Auth.

To sum up, Fig.7 shows the structure of Key_Share.

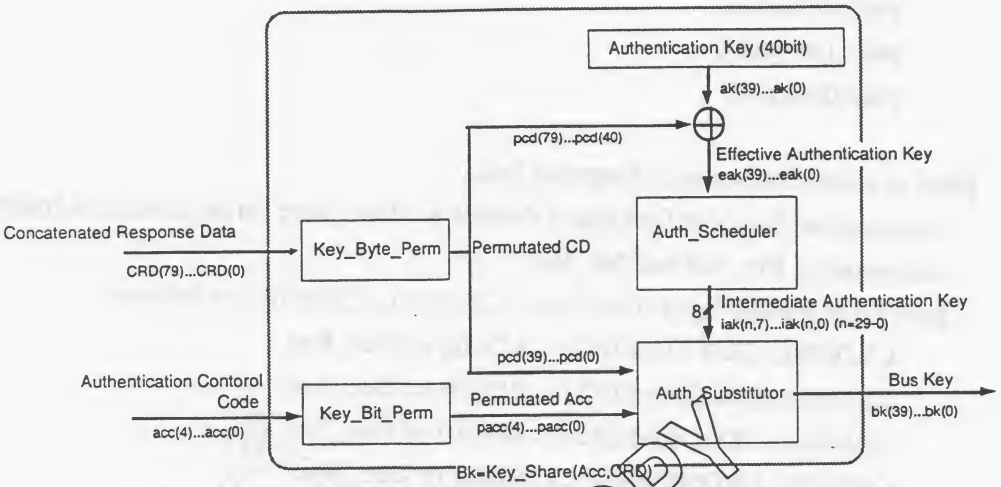


Fig.7 The structure of Key_Share

Next we define Key_Byte_Perm, Key_Bit_Perm. Note that the other parts are the same as ones in Drv_Auth explained in 4.2.

[Key_Byte_Perm]

Key_Byte_Perm permutates 80 bits Concatenated Response Data ( CRD(79)...CRD(0)) to 80 bits Permutated CD (pcd(79)...pcd(0)). The relation between input and output is as follows :

pcd(79)...pcd(72)=CRD(79)...CRD(72)
pcd(71)...pcd(64)=CRD(15)...CRD(8)
pcd(63)...pcd(56)=CRD(55)...CRD(48)
pcd(55)...pcd(48)=CRD(71)...CRD(64)
pcd(47)...pcd(40)=CRD(23)...CRD(16)
pcd(39)...pcd(32)=CRD(63)...CRD(56)
pcd(31)...pcd(24)=CRD(47)...CRD(40)
pcd(23)...pcd(16)=CRD(31)...CRD(24)
pcd(15)...pcd(8)=CRD(7)...CRD(0)
pcd(7)...pcd(0)=CRD(39)...CRD(32)

[Key_Bit_Perm]

Table2. Byte/Bit Order of Bus-Encrypted Secured Disc Key data

| | BP \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Block0 | 0 | bebd(39) | | | | | | | |
| | 1 | | | | | | | | |
| | 2 | | | | | | | | |
| | 3 | | | | | | | | |
| | 4 | | | | | | | | bebd(0) |
| Block1 | 5 | bebd(39) | | | | | | | |
| | 6 | | | | | | | | |
| | 7 | | | | | | | | |
| | 8 | | | | | | | | |
| | 9 | | | | | | | | bebd(0) |
| | ⋮ | | | | | | | | |
| Block408 | 2040 | bebd(39) | | | | | | | |
| | 2041 | | | | | | | | |
| | 2042 | | | | | | | | |
| | 2043 | | | | | | | | |
| | 2044 | | | | | | | | bebd(0) |
| Block409 | 2045 | bebd(39) | | | | | | | |
| | 2046 | | | | | | | | |
| | 2047 | | | | | | | | bebd(16) |



## 5. Usage of algorithm

Five algorithms are used in following order to realize Bus-Authentication and Bus-Decryption :

(1) Random number Generation

(2) Drv_Auth

(3) Dec_Auth : (2) or (3) algorithm uses the result of (1) as input.

(4) Key_Share : This algorithm uses the result of (2)(3) as input.

(5) Bus-Decrypt: This algorithm uses the result of (4) as input.

And the software controlling the authentication scheme using these algorithms shall be implemented in such a way general user other than authorized person who gains detain and confidential information of Content Scramble System can not reveal it.

Next we explain the usage of algorithms on DVD-ROM Drive in detail.

[ Drive authentication ]

Decoder Card authenticates DVD-ROM Drive as follows :

(1) Decoder Card generates 80 bits random number Drv_Chal, and sends it to DVD-ROM Drive.

To sum up, Fig.8 shows the Bus-Authentication/Decryption scheme of Decoder Card on Disc inserted, and Fig.9 shows the scheme before a playback of VTS.

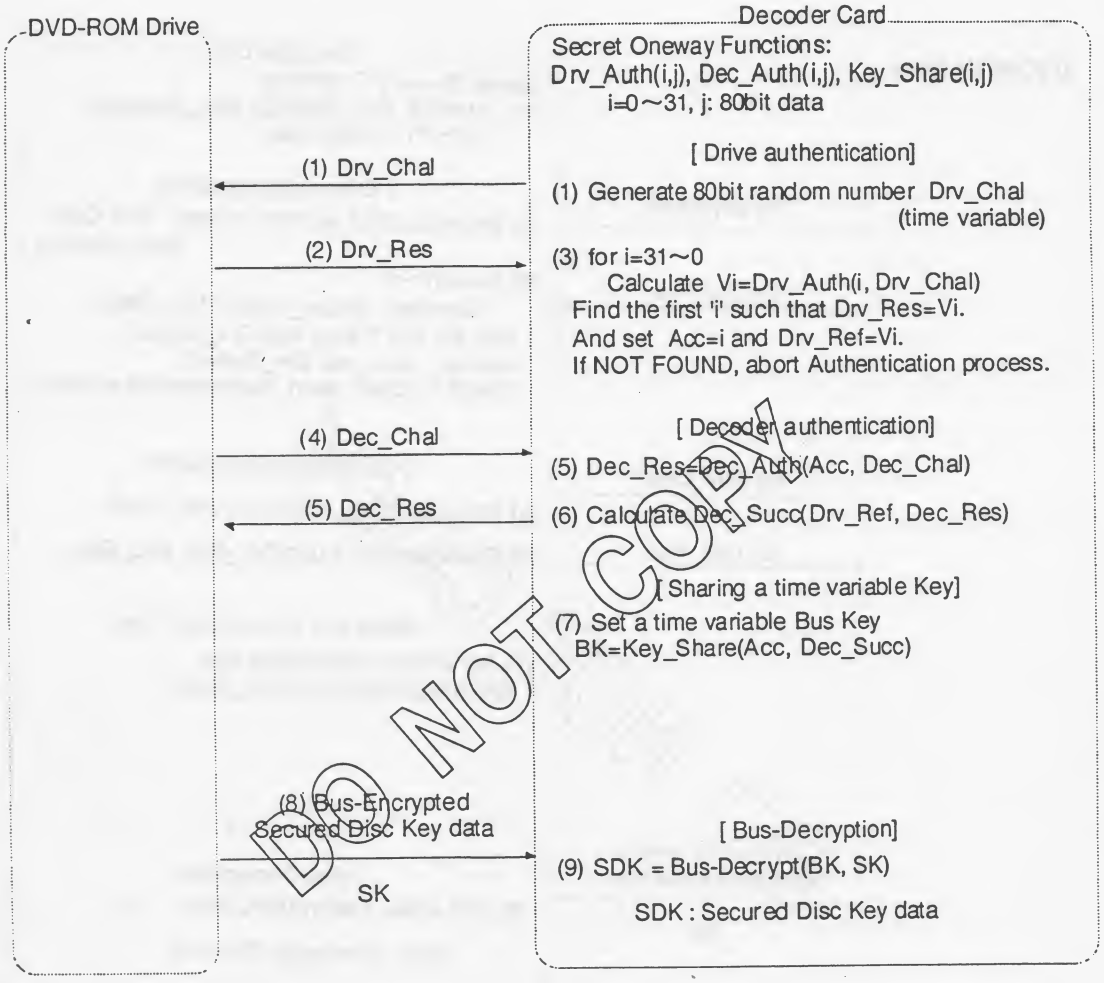




Fig.8. Bus-Authentication/Decryption scheme on Disc inserted

## 6. Implementation requirements

- Five algorithms (Random number Generation, Drv_Auth, Dec_Auth, Key_Share, Bus-Decrypt), Authentication Key, replaced ACC, Effective Authentication Key, and Intermediate Authentication Key shall be implemented in such a way general user other than authorized person who gains detail and confidential information of Content Scramble System can not reveal and/or modify them.
- Random number Generator should be a real random number generator or a Pseudo-random number generator whose linear complexity is over 41.
- The input of Drv_Auth in Authenticator of Decoder Card shall not be controlled from outside.
- The input of Key_Share in Authenticator shall not be controlled from outside.
- Any information except Input and Output of Authenticator on DVD-ROM Drive shall not be informed to users of Authenticator LSI.
- Switching to test mode shall not be done easily by the general user.
- The time variable Bus Key shall be concealed in such a way that general user other than the authorized person can not reveal it.
- Different Bus Key shall be used in each Bus-Decryption of Bus-Encrypted Secured Disc Key data and Bus-Encrypted Encrypted Title Key.

(a) Single Layer type
(b) Dual Layer type with Parallel Track Path
(c) Dual Layer type with Opposite Track Path

Each physical volume shall contain one Data Area and one or two Lead-in Area which shall have 192 Control Data Blocks. And, each Control Data Block shall consist of a Physical format information sector, a Disc manufacturing information sector and 14 Content provider information sectors. A physical sector shall have 2048 bytes of Main Data field and 6 bytes of Copyright Management Information field as a part of sector header which is described as CPR_MAI field in DVD Specifications.

The Copyright Management Information of this scheme shall consist of "Scramble Flag" which is described as CP_SEC flag in DVD Specifications, "Encrypted Title Key" and "Authentication Control Code".

In the Lead-in Area, "Secured Disc Key data (2048 bytes)" shall be recorded in the Main Data field of each first Content Provider information sector, which has a relative sector number 2. And "Authentication Control Code" shall be recorded in the CPR_MAI field of all Content Provider information sectors, which has a relative sector number from 2 to 15.

In the Data Area, both "Scramble Flag" field and "Encrypted Title Key (40 bits)" field are defined in the Copyright Management Information field of each sector. When a file is scrambled, the identical "Encrypted Title Key" shall be recorded in the CPR_MAI field of all sectors of a scrambled file. And, "Scramble Flag" shall be set to ONE in the Copyright Management Information field of all scrambled sectors. In case of non-scrambled sector, "Scramble Flag" shall be set to ZERO. When a file is not scrambled, "Scramble Flag" shall be set to ZERO and "Encrypted Title Key" field shall be reserved and set to ZERO in all sector of a file.
As for Navigation pack sectors, they shall belong to non-scrambled sectors.

An example of this data structure is shown in Fig. 2 for illustrative purposes only.

2.3.2 Data Structure in File System

A single volume space shall be defined on Data Area of each physical format. Both UDF file structure and ISO 9660 file structure shall be recorded on this volume space, and shall have the Copyright Management Information in the Implementation Use Extended Attribute of the File Entry for each file and in the System Use field of the Directory Record for each file respectively. The Copyright Management Information of these field shall contain CGMS Information, Data Structure Type and Protection System Information.

The CGMS Information shall have "Copyrighted Material" flag and CGMS code. When "Copyrighted Material" flag is set to ONE, the file contains Copyrighted Material including Audio and Video Object data and CGMS code of the file shall be interpreted as a 2 bits unsigned binary number as follows;

11b: No copying is permitted
10b: One generation copies may be made
01b: Condition is not used
00b: Copying is permitted without restrictions

Protection System Type shall be interpreted as an 8 bits unsigned binary number as follows;

0: This file has no scrambled sector
1: This file has one or more scrambled sectors that are processed according to this specification.
All other values are reserved for future standardization.



2.3.3 Data Structure in Packed Data

A DVD-VIDEO object set file shall consist of Navigation pack sectors, Video pack sectors, and optional Audio pack sectors, and Sub-picture pack sectors. When a file is scrambled, sectors of which "PES_Scrambling_control" is set to 01b may be scrambled on a sector by sector basis. Note that Navigation pack sector shall not be scrambled. The Main Data field except for the leading 128 bytes shall be scrambled in a scrambled sector.

"PES_scrambling_control" field in the packet header, which is defined in ISO/IEC 13818-1, is used to discriminate a "Scrambled sector" from a Non-scrambled sector" by PC software. In this specification, the value of this field is defined as follows;

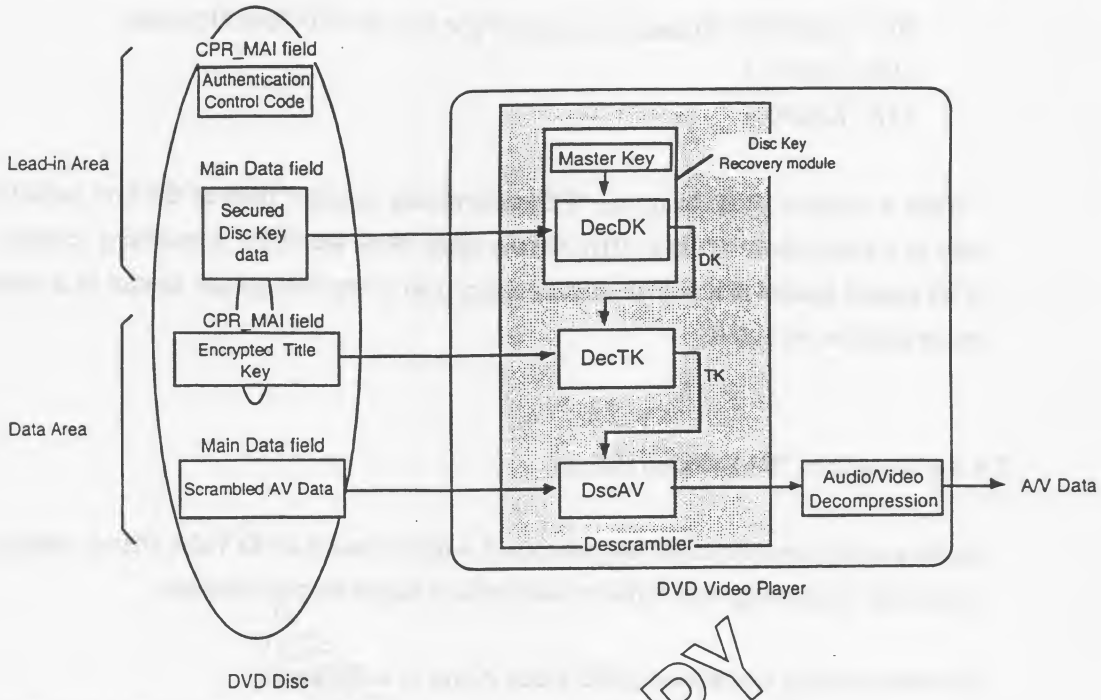


Fig.3 Architecture of "DVD Video Player System"

2.5 Architecture of "DVD Video Playback system on PC"

On PC system, it is necessary to append the "Bus-Authentication and Bus-Encryption between DVD-ROM Drive and Decoder Card" before descrambling the content.

The mutual Bus Authentication scheme between DVD-ROM Drive and Decoder Card is as follows:

(1) Decoder Card acts as "Verifier" and DVD-ROM Drive acts as "Prover". Verifier generates a random number, and sends it to another side as Challenge data.

(2) Prover calculates Response data using Authentication Control Code and secret oneway function, and returns it.

(3) Verifier receives Response data, and checks its validity. If the validity is not proven, the authentication process is aborted.

(4) Steps (1)-(3) repeat by changing the roles of Decoder Card and DVD-ROM Drive.

(Note)There are the plural oneway functions for calculating Response data. The Security Management Agent selects a oneway function. The information to select a function, called "Authentication Control Code", is stored in Lead-in area of Disc.

As the result of the successful mutual authentication scheme, both sides generate the same time-variable key (it is denoted by Bus Key or simply 'BK').





 

## Index

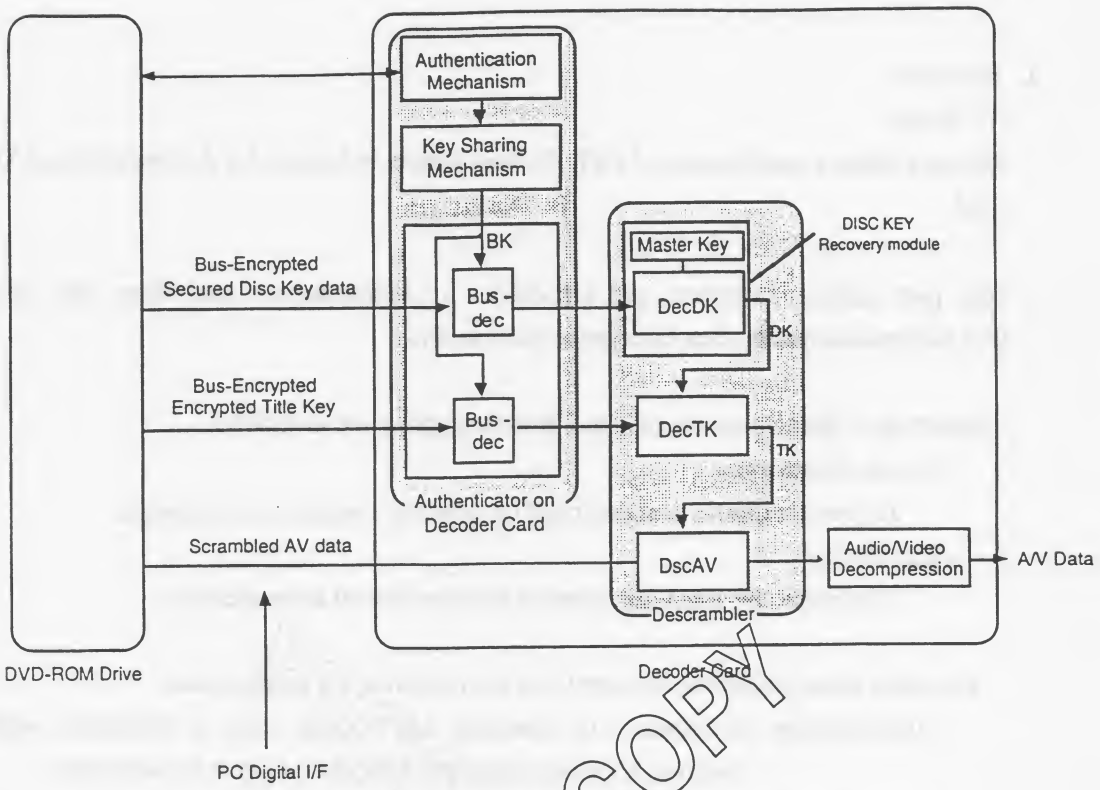


Fig.1. Architecture of Decoder Card



- Functions and algorithms in Authenticator
  - Random number Generation : Generate a random challenge data
  - Drv_Auth : Oneway function for Drive authentication, which is Decoder Card's act of authenticating the validity of DVD-ROM Drive.
  - Dec_Auth : Oneway function for Decoder authentication, which is DVD-ROM Drive's act of authenticating the validity of Decoder Card.
  - Key_Share : Oneway function for key sharing between DVD-ROM Drive and Decoder Card
  - Bus-Decrypt : Bus-Encrypted Secured Disc Key data and Bus-Encrypted Encrypted Title Key are decrypted by the shared key.

- Drv_Auth, Dec_Auth and Key_Share
  - Using oneway function

    By using secret oneway function, it is practically impossible to guess Authentication Key and its algorithm from its input and output of Authenticator. Each oneway function has oneway operation T-box.
  - 32 types of triplet for each Drv_Auth/Dec_Auth/Key_Share (k=0-31)

    To keep the security high, the oneway functions used in Authentication process

And ⊕ means "Exclusive OR" operation.

## 2. Input/Output

We define the input and output of five algorithms in Authenticator on Decoder Card.

Note that there are several internal input to the algorithm. All inputs and outputs without (internal) marks are external.

- Random number Generation
  - [Output] Drv_Chal (80 bits) : Authenticator_on_Decoder's Challenge data for Drive authentication.
- Drv_Auth
  - [Input] Authentication Control Code (5 bits)
  - [Output] Drv_Ref (40 bits) : Authenticator_on_Decoder's Reference data for Drive authentication.
  - [Input(Internal)] Drv_Chal made by Random number Generator
- Dec_Auth
  - [Input] Dec_Chal (80 bits) : Authenticator_on_Drive's Challenge data for Decoder authentication.
  - [Input] Authentication Control Code (5 bits)
  - [Output] Dec_Res (40 bits) : Authenticator_on_Decoder's Response data for Decoder authentication.
- Key_Share
  - [Input] Authentication Control Code (5 bits)
  - [Input(Internal)] Dec_Succ (80 bits), which is a concatenation of internally stored Drv_Ref (40 bits) and internally stored Dec_Res (40 bits)
  - [Output(Internal)] Bus Key (40 bits) shall be used to Bus-Decryption directly.
- Bus-Decrypt
  - <On Disc Inserted>
    - [Input] Bus-Encrypted Secured Disc Key data by BK (2048 bytes)
    - [Output] Connect to Descrambler without appearing outside
  - <Before a playback of VTS>
    - [Input] Bus-Encrypted Encrypted Title Key (40 bits)
    - [Output] Connect to Descrambler without appearing outside

## 3. Process

### 3.1 Process of Random number Generation

Random number Generation generates 80 bits random number to be used for a challenge to another side.

DVD-ROM Drive Authentication algorithm "Drv_Auth" is used to calculate Reference Data "Drv_Ref" from Challenge Data "Drv_Chal".

Drv_Auth has 5 bits Authentication Control Code "ACC" and 80 bits Drv_Chal as input, and 40 bits Drv_Ref as output. Each bit of ACC is denoted as acc(4)...acc(0), each bit of Drv_Chal is CD(79)...CD(0), and each bit of Drv_Ref is RD(39)...RD(0).

Drv_Auth consists of the following parts :

- Drv_Byte_Perm

  Each byte of 80 bits Drv_Chal is permutated to Permuted CD.

  Each bit of Drv_Chal is denoted as CD(79)...CD(0), and each bit of Permutated CD is denoted as pcd(79)...pcd(0).

- Drv_Bit_Perm

  Each bit of 5 bits ACC is permutated to Permutated ACC.

  Each bit of ACC is denoted as acc(4)...acc(0), and each bit of Permutated ACC is denoted as pacc(4)...pacc(0).

- Authentication Key

  Authentication Key ak(39)...ak(0) is 40 bits length, and it is embedded in Drv_Auth algorithm. Authentication Key is shown in Annex-A.

- EOR

  Authentication Key is EORed with pcd(79)...pcd(40). The result is called Effective Authentication Key denoted as eak(39)...eak(0).

- Auth_Scheduler

  It is initialized by Effective Authentication Key eak(39)...eak(0), and outputs every 8 bits Intermediate Authentication Key which is denoted as iak(n,7)...iak(n,0) for each n=29-0. Thus total 30 bytes Intermediate Authentication Key is generated.

- Auth_Substitutor

  It calculates Drv_Ref (RD(39)...RD(0)) using Permutated CD (pcd(39)...pcd(0)) , Permutated ACC (pacc(4)...pacc(0)) and Intermediate Authentication Key (iak(n,7)...iak(n,0) for n=29-0 ) .

To sum up, Fig.2 shows the structure of Drv_Auth.

pacc(4)=acc(4)
pacc(3)=acc(3)
pacc(2)=acc(2)
pacc(1)=acc(1)
pacc(0)=acc(0)

(Note) Drv_Bit_Perm operates nothing actually, but we dare to define in order to contrast this Drv_Bit_Perm with Dec_Bit_Perm in "Dec_Auth "and Key_Bit_Perm in "Key_Share".

[Auth_Scheduler]

Auth_Scheduler prepares every 8 bits Intermediate Authentication Key for n=29-0 using 40 bits Effective Authentication Key 'eak'. Auth_Scheduler is initialized by 40 bits eak(39)...eak(0) and outputs every 8 bits iak(n,7)...iak(n,0) for n=29-0 by operating 30×8 times.

Auth_Scheduler consists of the following parts :

- Linear Feedback Shift Register of 17 degree (Upper_LFSR)
  The polynomial of the Upper_LFSR is $x^{17}+x^3+1$.
- Linear Feedback Shift Register of 25 degree (Lower_LFSR)
  The polynomial of the Lower_LFSR is $x^{25}+x^{22}+x^{21}+x^{13}+1$.
- LSW=1
  Reversing the output of Lower_LFSR
- USW=1
  Reversing the output of Upper_LFSR
- 1 bit Full adder with carry
  Two input bits and feedbacked carry bit are full-added.
  The lower bit of the result is a bit of Intermediate Authentication Key, and the upper bit is stored to D-flip flop (DFF) for the next bit addition.

Auth_Scheduler is initialized by 40 bits Effective Authentication Key eak(39)...eak(0) as follows :

- initialize of Upper_LFSR
  Bit 8 is preset to "1".
  eak(31)...eak(24) are set to Bit16...Bit9, and eak(39)...eak(32) are set to Bit7...Bit0.
- Initialization of Lower_LFSR
  Bit 3 is preset to "1".
  eak(7)...eak(0) are set to Bit24...Bit17 and eak(15)...eak(8) are set to Bit16...Bit9 and eak(23)...eak(19) are set to Bit8...Bit4 and eak(18)...eak(16) are set to Bit2...Bit0.

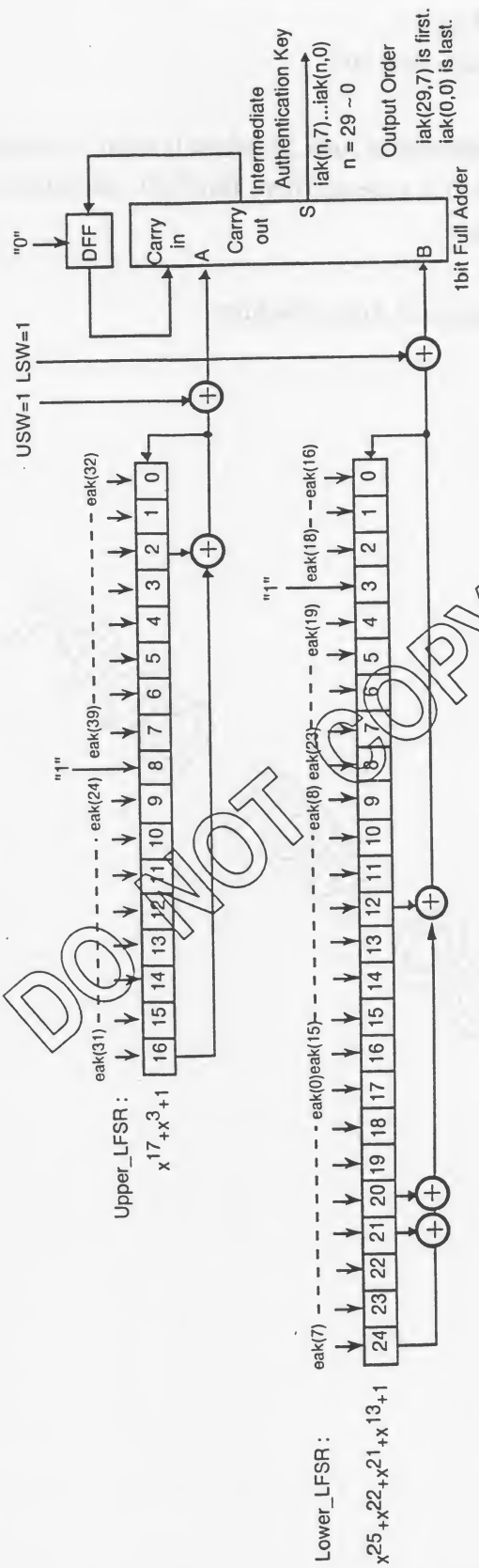


Fig.3 The structure of Auth_Scheduler

⊕temp2(39)...temp2(32)⊕temp2(7)...temp2(0))
temp3(23)...temp3(16)=T(Auth_Sbox(iak(17,7)...iak(17,0),
temp2(23)...temp2(16)) ⊕ temp2(31)...temp2(24))
temp3(15)...temp3(8)=T(Auth_Sbox(iak(16,7)...iak(16,0),
temp2(15)...temp2(8)) ⊕ temp2(23)...temp2(16))
temp3(7)...temp3(0)=T(Auth_Sbox(iak(15,7)...iak(15,0), temp2(7)...temp2(0))
⊕ temp2(15)...temp2(8))

<4th round>
temp4(39)...temp4(32)=T(Auth_Sbox(iak(14,7)...iak(14,0),
temp3(39)...temp3(32)⊕temp3(7)...temp3(0)) )
temp4(31)...temp4(24)=T(Auth_Sbox(iak(13,7)...iak(13,0),
temp3(31)...temp3(24))
⊕ temp3(39)...temp3(32)⊕temp3(7)...temp3(0))
temp4(23)...temp4(16)=T(Auth_Sbox(iak(12,7)...iak(12,0),
temp3(23)...temp3(16)) ⊕ temp3(31)...temp3(24))
temp4(15)...temp4(8)=T(Auth_Sbox(iak(11,7)...iak(11,0),
temp3(15)...temp3(8)) ⊕ temp3(23)...temp3(16))
temp4(7)...temp4(0)=T(Auth_Sbox(iak(10,7)...iak(10,0), temp3(7)...temp3(0))
⊕ temp3(15)...temp3(8))

<5th round>
temp5(39)...temp5(32)=Auth_Sbox(iak(9,7)...iak(9,0), temp4(39)...temp4(32)
⊕temp4(7)...temp4(0))
temp5(31)...temp5(24)=Auth_Sbox(iak(8,7)...iak(8,0), temp4(31)...temp4(24))
⊕ temp4(39)...temp4(32)⊕temp4(7)...temp4(0)
temp5(23)...temp5(16)=Auth_Sbox(iak(7,7)...iak(7,0), temp4(23)...temp4(16))
⊕ temp4(31)...temp4(24)
temp5(15)...temp5(8)=Auth_Sbox(iak(6,7)...iak(6,0), temp4(15)...temp4(8))
⊕ temp4(23)...temp4(16)
temp5(7)...temp5(0)=Auth_Sbox(iak(5,7)...iak(5,0), temp4(7)...temp4(0))
⊕ temp4(15)...temp4(8)

<6th round>
RD(39)...RD(32)=Auth_Sbox(iak(4,7)...iak(4,0), temp5(39)...temp5(32)
⊕temp5(7)...temp5(0))
RD(31)...RD(24)=Auth_Sbox(iak(3,7)...iak(3,0), temp5(31)...temp5(24))
⊕ temp5(39)...temp5(32)⊕temp5(7)...temp5(0)
RD(23)...RD(16)=Auth_Sbox(iak(2,7)...iak(2,0), temp5(23)...temp5(16))

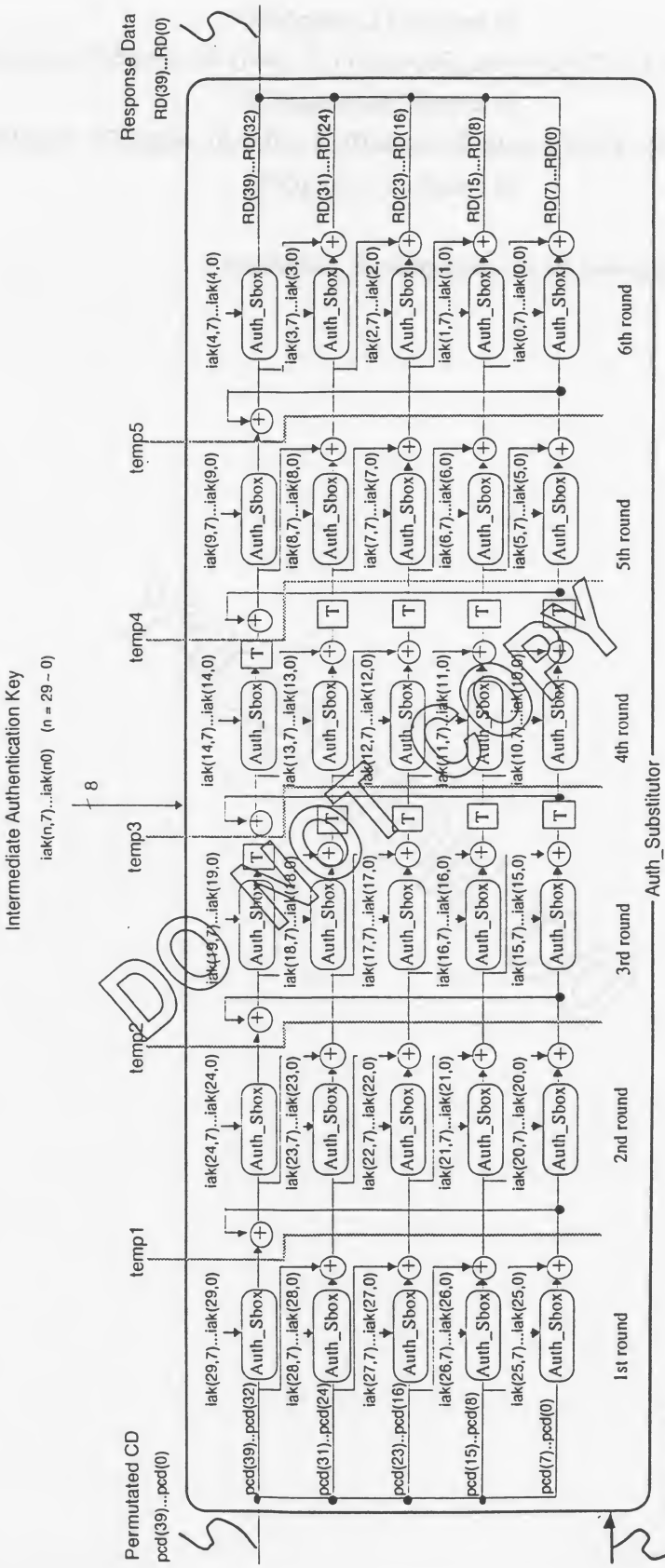


Fig.4 Structure of Auth_Substitutor



 

OUT(3)=IN(3) ⊕ IN(4)
OUT(4)=IN(4) ⊕ IN(5)
OUT(5)=IN(5) ⊕ IN(6)
OUT(6)=IN(6) ⊕ IN(7)
OUT(7)=IN(7) ⊕ IN(0)

4.3 Decoder Card Authentication algorithm--- Dec_Auth

Decoder Card Authentication algorithm "Dec_Auth" is used to calculate Response Data "Dec_Res" from Challenge Data "Dec_Chal".

Dec_Auth has 5 bits Authentication Control Code "ACC", 80 bits Dec_Chal as input, and 40 bits Dec_Res as output. Each bit of ACC is denoted as acc(4)...acc(0), each bit of Dec_Chal is CD(79)...CD(0), each bit of Dec_Res is RD(39)...RD(0).

Dec_Auth consists of following parts :

- Dec_Bytes_Perm

  Each byte of 80 bits Dec_Chal is permutated to Permutated CD.
  Each bit of Dec_Chal is denoted as CD(79)...CD(0), and each bit of Permutated CD is denoted as pcd(79)...pcd(0).

- Dec_Bit_Perm

  Each bit of 5 bits ACC is permutated to Permutated ACC.
  Each bit of ACC is denoted as acc(4)...acc(0), and each bit of Permutated ACC is denoted as pacc(4)...pacc(0).

- Authentication Key

  A Secret Authentication Key ak(39)...ak(0) is 40 bits length, and it is the same as one in Drv_Auth.

- EOR

  Authentication Key is EORed with pcd(79)...pcd(40). The result is Effective Authentication Key eak(39)...eak(0). This part is the same as one in Drv_Auth.

- Auth_Scheduler

  It is initialized by Effective Authentication Key eak(39)...eak(0), and outputs 30 bytes Intermediate Authentication Key which is denoted as iak(n,7)...iak(n,0) for n=29-0.
  This part is the same as one in Drv_Auth.

- Auth_Substitutor

  It calculates Drv_Res (RD(39)...RD(0)) using Permutated CD (pcd(39)...pcd(0)) , Permutated ACC (pacc(4)...pacc(0)) and Intermediate Authentication Key (iak(n,7)...iak(n,0) for n=29-0 ) . This part is the same as one in Drv_Auth.

Dec_Bit_Perm permutates 5 bits input (acc(4)...acc(0)) to 5 bits output (pacc(4)...pacc(0)). The relation between input bit and output bit is as follows :

pacc(4)=acc(3)
pacc(3)=~acc(4) '~' means negation.
pacc(2)=acc(1)
pacc(1)=~acc(0)
pacc(0)=acc(2)

4.4 Key sharing algorithm--- Key_Share

Key sharing algorithm "Key_Share" is used to calculate sharing key "Bus Key" from Concatenated Response Data on both Decoder and Drive :

Key_Share has 5 bits Authentication Control Code "ACC", 80 bits Concatenated Response Data as input, and 40 bits Bus Key as output. We define how to create Concatenated Response Data later. Each bit of ACC is denoted as acc(4)...acc(0), each bit of Concatenated Response Data is CRD(79)...CRD(0), each bit of Bus Key is bk(39)...bk(0).

"Key_Share" consists of following parts :

- Key_Byte_Perm

  Each byte of 80 bits Concatenated Response Data is permutated to Permutated CD. Each bit of Concatenated Response Data is denoted as CRD(79)...CRD(0), and each bit of Permutated CD is denoted as pcd(79)...pcd(0).

- Key_Bit_Perm

  Each bit of 5 bits ACC is permutated to Permutated ACC.
  Each bit of ACC is denoted as acc(4)...acc(0), and each bit of Permutated ACC is denoted as pacc(4)...pacc(0).

- Authentication Key

  Authentication Key ak(39)...ak(0) is 40 bits length, and it is the same as one in Drv_Auth.

- EOR

  Authentication Key is EORed with pcd(79)...pcd(40). The result is Effective Authentication Key eak(39)...eak(0). This part is the same as one in Drv_Auth.

- Auth_Scheduler

  It is initialized by Effective Authentication Key eak(39)...eak(0), and outputs 30 bytes Intermediate Authentication Key which is denoted as iak(n,7)...iak(n,0) for n=29-0.
  This part is the same as one in Drv_Auth.

- Auth_Substitutor

  It calculates Bus Key (bk(39)...bk(0)) using Permutated CD (pcd(39)...pcd(0)) ,

Key_Bit_Perm permutates 5 bits input (acc(4)...acc(0)) to 5 bits output (pacc(4)...pacc(0)).
The relation between input bit and output bit is as follows :

pacc(4)=~acc(2) '~' means negation
pacc(3)=acc(0)
pacc(2)=acc(1)
pacc(1)=~acc(3)
pacc(0)=acc(4)

[How to create Concatenated Response Data]

Concatenated Response Data that is denoted as "Dec_Succ" in the protocol is created by concatenating Drv_Ref and Dec_Res.

Each bit of internal input "Dec_Succ" (CRD(79)...CRD(0)) is as follows :

CRD(79)...CRD(72)=RD(39)...RD(32) of Dec_Res
CRD(71)...CRD(64)=RD(31)...RD(24) of Dec_Res
CRD(63)...CRD(56)=RD(23)...RD(16) of Dec_Res
CRD(55)...CRD(48)=RD(15)...RD(8) of Dec_Res
CRD(47)...CRD(40)=RD(7)...RD(0) of Dec_Res

CRD(39)...CRD(32)=RD(39)...RD(32) of Drv_Ref
CRD(31)...CRD(24)=RD(31)...RD(24) of Drv_Ref
CRD(23)...CRD(16)=RD(23)...RD(16) of Drv_Ref
CRD(15)...CRD(8)=RD(15)...RD(8) of Drv_Ref
CRD(7)...CRD(0)=RD(7)...RD(0) of Drv_Ref



4.5 Bus-Decryption algorithm---Bus-Decrypt

On Disc Inserted, 2048 bytes of Bus-Encrypted Secured Disc Key data is divided into 409 of 5 bytes blocks (Block0,...Block408 from the top) and a 3 bytes block (Block409). Each 5 bytes block is denoted by bebd(39)...bebd(0). Table2 shows Byte/Bit Order of Bus-Encrypted Secured Disc Key data.

Each Bus-Encrypted Block data is EORed with the time variable key BK (it is denoted as bk(39)...bk(0)). The result, Block data, is described as

bd(39)...bd(0) = bebd(39)...bebd(0) ⊕ bk(39)...bk(0).

Universal BK is used to decrypt every 40 bits of Bus-Encrypted Secured Disc Key data. Bus-Decryption of the Block409 which has 3 bytes block data is arbitrary.

Before a playback of VTS, 40 bits of Bus-Encrypted Encrypted Title Key, beetk(39)...beetk(0), is EORed with the BK. The result, Encrypted Title Key, is described as

etk(39)...etk(0) = beetk(39)...beetk(0) ⊕ bk(39)...bk(0).

(2) Decoder Card receives response data Drv_Res from DVD-ROM Drive.

(3) Decoder Card calculates Vi=Drv_Auth(i, Drv_Chal) for i=31-0, and finds the first 'i' such that Drv_Res=Vi. And sets ACC=i, and Drv_Ref=Vi. If not found, abort Authentication process.

[ Decoder authentication ]

DVD-ROM Drive authenticates Decoder Card as follows :

(4) Decoder Card receives 80 bits random number Dec_Chal from DVD-ROM Drive.

(5) Decoder Card calculates response data such as Dec_Res=Dec_Auth(ACC, Dec_Chal) using the Drv_Auth function indicated by ACC, and sends back to DVD-ROM Drive.

(6) Decoder Card calculates Concatenated Response Data (Dec_Succ) using Dec_Res and Drv_Ref as in 4.4.

DVD-ROM Drive authenticates Decoder Card using Dec_Chal and the corresponding Dec_Res.

[ Sharing a time variable key ]

(7) Decoder Card calculates BK=Key_Share(ACC, Dec_Succ) using 'ACC' in (3).

After (1)-(7), proper DVD-ROM Drive and Decoder Card pair can share the same time variable key BK.



[ Bus-Decryption ]

- On Disc inserted

(8) Decoder Card receives Bus-Encrypted Secured Disc Key data SK from DVD-ROM Drive.

(9) Decoder Card decrypts Bus-Encrypted Secured Disc Key data using BK and obtains Secured Disc Key data (2048 bytes).

- Before a playback of VTS

(8) Decoder Card receives Bus-Encrypted Encrypted Title Key SK from DVD-ROM Drive.

(9) Decoder Card decrypts Bus-Encrypted Encrypted Title Key using BK and obtains Encrypted Title Key (40 bits).

After the process explained above, Decoder Card gets Secured Disc Key data (2048 bytes), Encrypted Title Key (40 bits) and Scrambled AV data.

Then Decoder Card operates the following Descramble process using them.

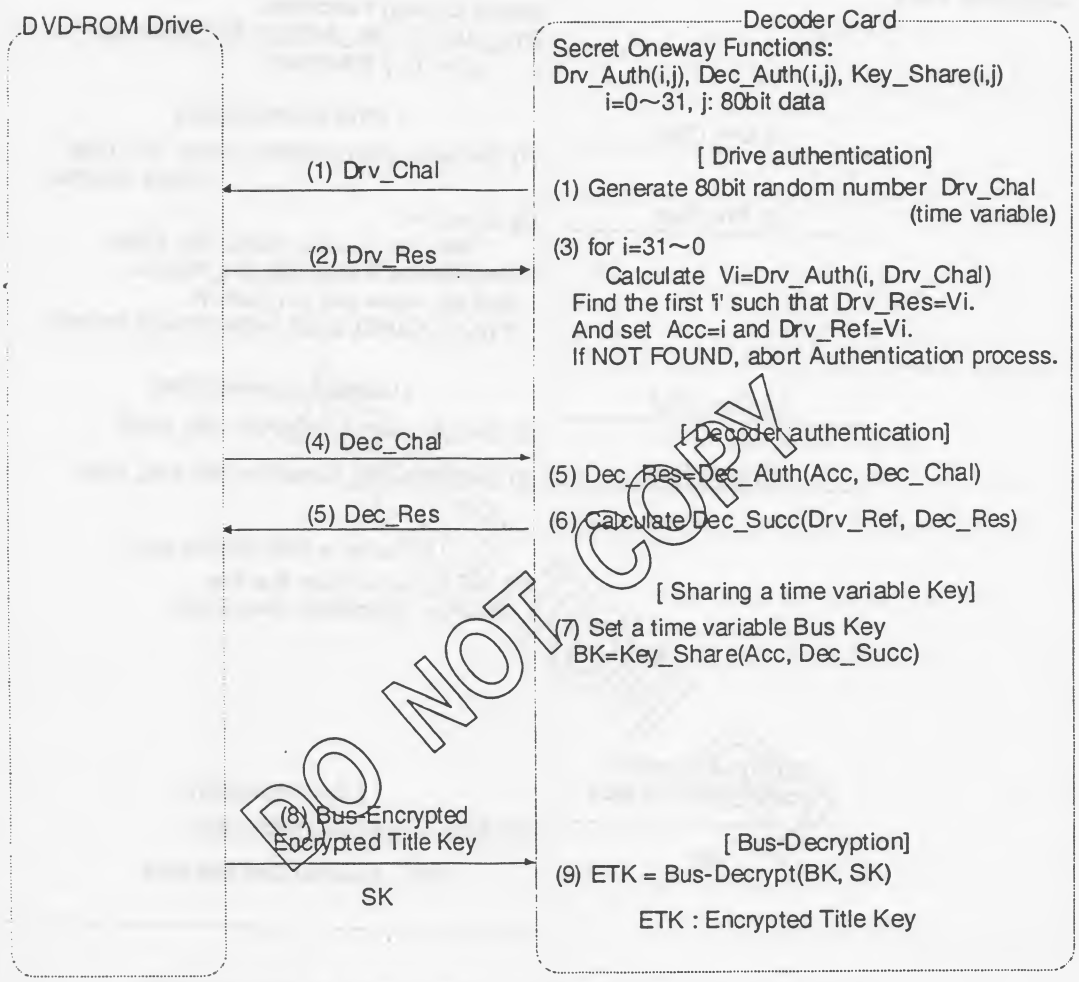




Fig.9. Bus-Authentication/Decryption scheme before a playback of VTS

(Annex-A) Authentication Key

This part is intentionally left blank.

(Annex_B) Pre-Table

This part is intentionally left blank.

(Annex_C) Post-Table

This part is intentionally left blank.